SHLOK SHIVARAM IYER SECTION:3E 1BM22CS260

1. 8) Write a program

a) To construct a binary Search tree.

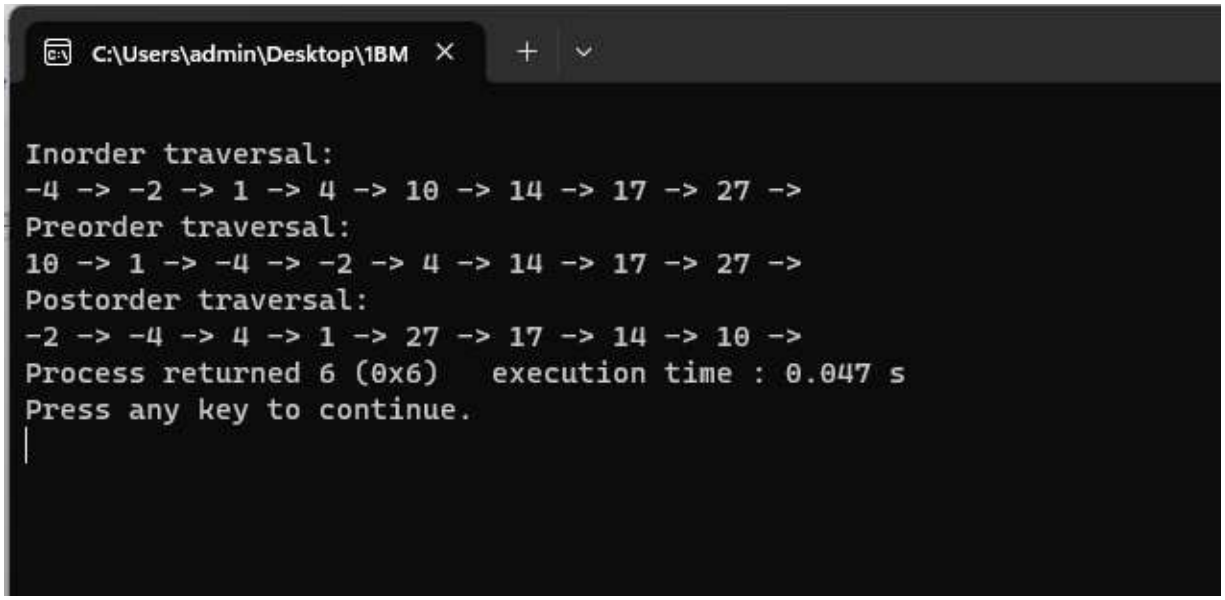
b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

```
#include <stdio.h>
#include <stdlib.h>

struct node {
  int key;
  struct node *left, *right;
};


struct node *newNode(int item) {
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->key = item;
  temp->left = temp->right = NULL;
  return temp;
}


void inorder(struct node *root) {
  if (root != NULL) {

    inorder(root->left);

    printf("%d -> ", root->key);


    inorder(root->right);
  }
}
```

```
void preorder(struct node *root) {
  if (root != NULL) {

    printf("%d -> ", root->key);

    preorder(root->left);

    preorder(root->right);
  }
}


void postorder(struct node *root) {
  if (root != NULL) {

    postorder(root->left);

    postorder(root->right);

    printf("%d -> ", root->key);
  }
}


struct node *insert(struct node *node, int key) {

  if (node == NULL) return newNode(key);


  if (key < node->key)
    node->left = insert(node->left, key);
  else
    node->right = insert(node->right, key);

  return node;
}


struct node *minValueNode(struct node *node) {
  struct node *current = node;

  while (current && current->left != NULL)
```

```
    current = current->left;

  return current;
}


int main() {
  struct node *root = NULL;
  root = insert(root, 10);
  root = insert(root, 1);
  root = insert(root, 14);
  root = insert(root, 17);
  root = insert(root, 27);
  root = insert(root, 4);
  root = insert(root, -4);
  root = insert(root, -2);

  printf("\nInorder traversal: \n");
  inorder(root);

  printf("\nPreorder traversal: \n");
  preorder(root);

  printf("\nPostorder traversal: \n");
  postorder(root);

}
```



2. 9a) Write a program to traverse a graph using the BFS method.

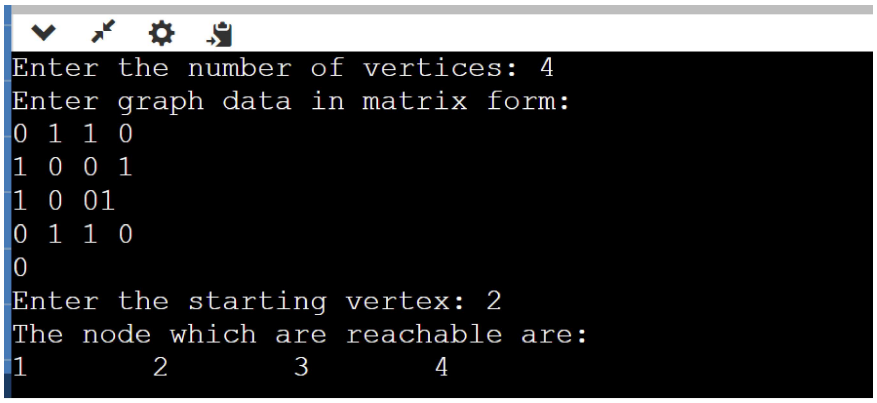
9b) Write a program to check whether a given graph is connected or not using the DFS method.

a) BSF

```
#include <stdio.h>
 int n, i, j, visited[10], queue[10], front = -1, rear = -1;
int adj[10][10];

void bfs(int v)
{
   for (i = 1; i <= n; i++)
      if (adj[v][i] && !visited[i])
         queue[++rear] = i;
   if (front <= rear)
   {
      visited[queue[front]] = 1;
      bfs(queue[front++]);
   }
}

void main()
{
   int v;
   printf("Enter the number of vertices: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++)
```

```
    {
        queue[i] = 0;
        visited[i] = 0;
    }
    printf("Enter graph data in matrix form:    \n");
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            scanf("%d", &adj[i][j]);
    printf("Enter the starting vertex: ");
    scanf("%d", &v);
    bfs(v);
    printf("The node which are reachable are:    \n");
    for (i = 1; i <= n; i++)
        if (visited[i])
            printf("%d\t", i);
        else
            printf("BFS is not possible. Not all nodes are reachable");

}
```

OUTPUT:



b)DSF

b) DFS

```
#include<stdio.h>
#include<conio.h>
int a[20][20], reach[20], n;
void dfs(int v) {
   int i;
   reach[v] = 1;
   for (i = 1; i <= n; i++)
      if (a[v][i] && !reach[i]) {
         printf("\n %d->%d", v, i);
         dfs(i);
      }
}

int main(int argc, char **argv) {
   int i, j, count = 0;
   printf("\n Enter number of vertices:");
   scanf("%d", &n);
```

```
    for (i = 1; i <= n; i++) {
        reach[i] = 0;
        for (j = 1; j <= n; j++)
            a[i][j] = 0;
    }
    printf("\n Enter the adjacency matrix:\n");
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            scanf("%d", &a[i][j]);
    dfs(1);
    printf("\n");
    for (i = 1; i <= n; i++) {
        if (reach[i])
            count++;
    }
    if (count == n)
        printf("\n Graph is connected");
    else
        printf("\n Graph is not connected");
    return 0;
}
```

OUTPUT:

```
Enter number of vertices:4

Enter the adjacency matrix:
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0

1->2
2->4
4->3

Graph is connected
```

```
Enter number of vertices:4

Enter the adjacency matrix:
1 0 0 0
0 0 0 0
0 0 1 1
0 0 1 1


Graph is not connected
```